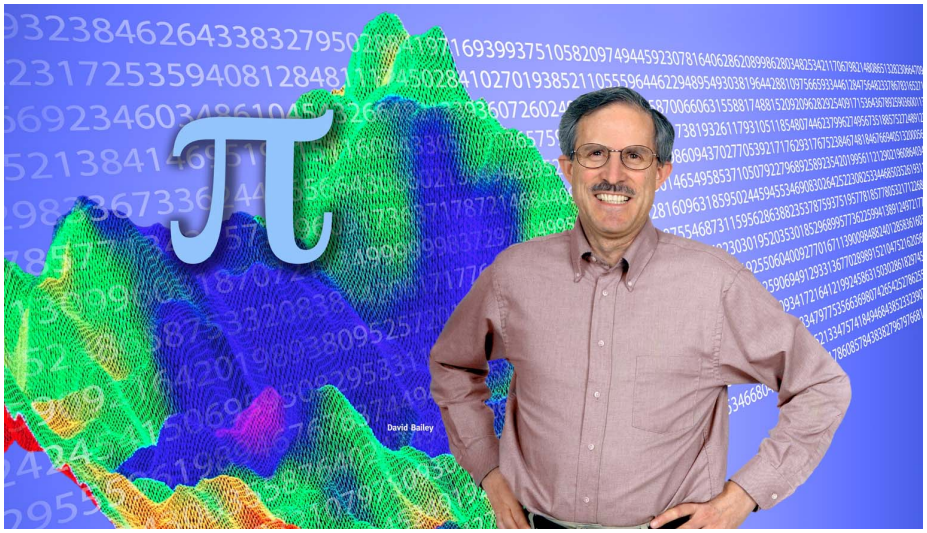
High-Precision Computation: Mathematical Physics and Dynamics

David H Bailey, Lawrence Berkeley National Lab, USA (speaker)

http://crd.lbl.gov/~dhbailey

Roberto Barrio, University of Zaragoza, Spain

Jonathan M. Borwein, University of Newcastle, Australia



Aren't 64 (or 80) bits enough?

Almost all scientific computers (from PCs to supercomputers) now feature IEEE-754 64-bit floating-point arithmetic. Many systems feature 80-bit hardware, although results are typically rounded to 64-bits before storing.

However, for a growing body of numerical algorithms and applications, 64 (or 80) bits aren't enough:

- Algorithms involving linear systems with large condition numbers.
- Summations of series with oscillating terms and many cancellations.
- Iterative calculations with many steps.
- Very highly parallel computations, which greatly exacerbate numerical difficulties.
- Studies in experimental mathematics – hundreds or thousands of digits.

Increasingly, researchers and engineers who develop and use codes are not experts in numerical analysis and may not realize the potential difficulties.

Using high-precision arithmetic is often the easiest way to solve numerical problems, even if other, more sophisticated solutions are possible.

Innocuous-looking example where high precision is required

Consider this simple problem: Find a polynomial function to fit the data (5, 2304, 118101, 1838336, 14855109, 79514880, 321537749, 1062287616, 3014530821).

The usual approach is to solve the linear system:

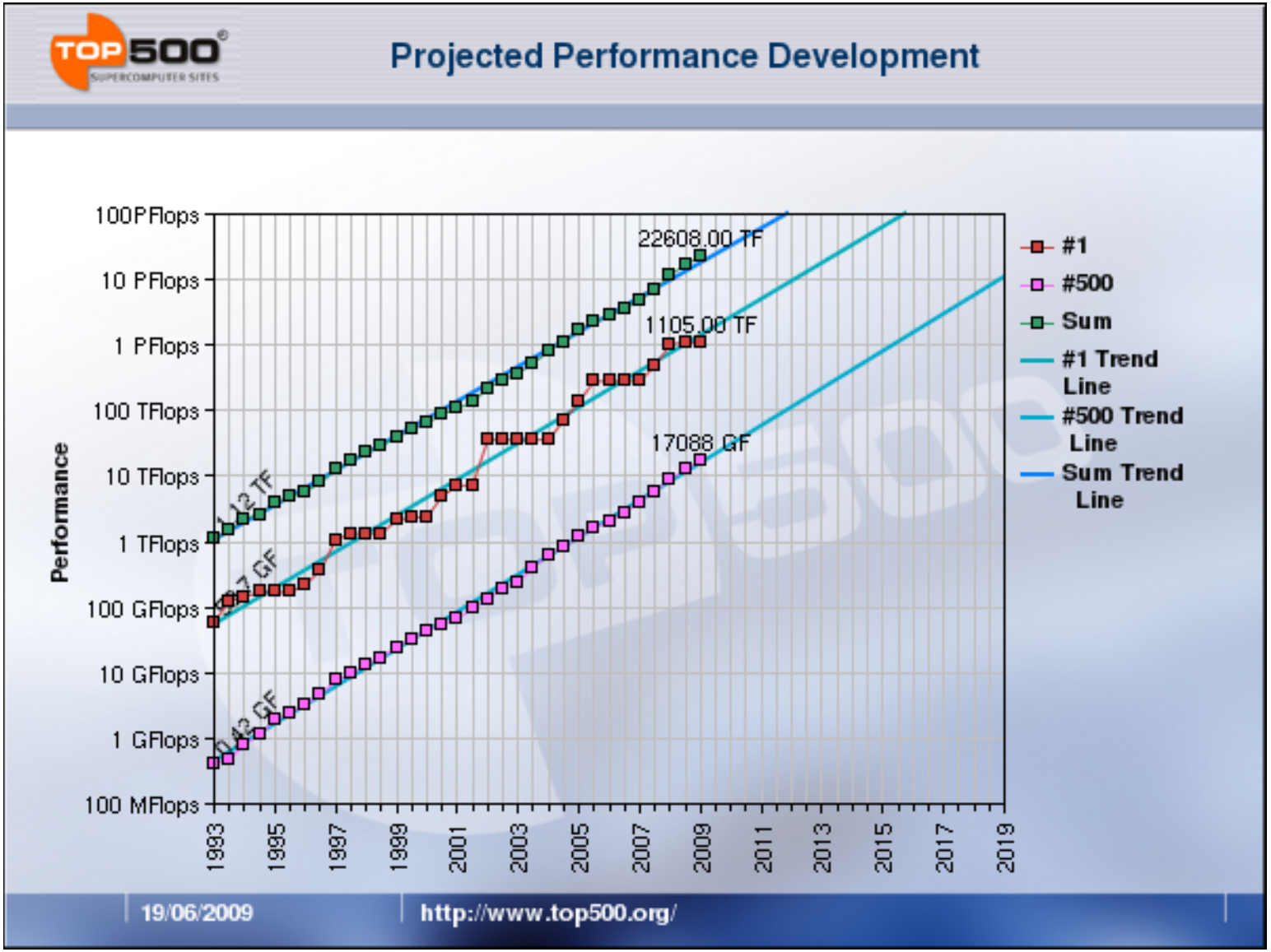
$$\begin{bmatrix} n & \sum_{k=1}^{n} x_k & \cdots & \sum_{k=1}^{n} x_k^n \\ \sum_{k=1}^{n} x_k & \sum_{k=1}^{n} x_k^2 & \cdots & \sum_{k=1}^{n} x_k^{n+1} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{k=1}^{n} x_k^n & \sum_{k=1} x_k^{n+1} & \cdots & \sum_{k=1}^{n} x_k^{2n} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} \sum_{k=1}^{n} y_k \\ \sum_{k=1}^{n} x_k y_k \\ \vdots \\ \sum_{k=1}^{n} x_k^n y_k \end{bmatrix}$$

using Matlab, Linpack or LAPACK. A computation with 64-bit (or 80-bit) floating-point arithmetic fails to find the correct result.

However, if, say, the Linpack routines are converted to use double-double arithmetic (approx. 31-digit accuracy), the above computation quickly concludes that the input data is exactly given by the simple formula:

$$f(k) = 5 + 220k^2 + 990k^4 + 924k^6 + 165k^8$$

Progress of Scientific Supercomputers: Data from the Top500 List



Very highly parallel computations exacerbate numerical difficulties

Consider the very simple 1-D differential equation $y''(x) = -f(x)$ for some function $f(x)$.

Discretization of this system immediately leads to the matrix

$$\begin{bmatrix} 2 & -1 & 0 & 0 & \cdots & 0 \\ -1 & 2 & -1 & 0 & \cdots & 0 \\ 0 & -1 & 2 & -1 & \cdots & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & \cdots & -1 & 2 & -1 & 0 \\ 0 & \cdots & 0 & -1 & 2 & -1 \\ 0 & \cdots & 0 & 0 & -1 & 2 \end{bmatrix}$$

whose condition number, for large n, is approximated by

$$\kappa(n) \approx \frac{4(n+1)^2}{\pi^2}$$

For the huge values of n now being used in highly parallel calculations, $\kappa(n)$ is large enough that results (depending on the function $f(x)$) may not be numerically reliable.

Free software for high-precision computation

- ARPREC. Arbitrary precision, with many algebraic and transcendental functions. High-level interfaces for C++ and Fortran-90 – existing codes can be converted with only minor effort. Available at http://crd.lbl.gov/~dhbailey/mpdist.
- GMP. Produced by a volunteer effort and is distributed under the GNU license by the Free Software Foundation. Available at http://gmplib.org.
- MPFR. C library for multiple-precision floating-point computations with exact rounding, based on GMP. Available at http://www.mpfr.org.
- MPFR++. High-level C++ interface to MPFR. Available at http://perso.ens-lyon.fr/nathalie.revol/software.html.
- GMPFRXX. Similar to MPFR++. Available at http://math.berkeley.edu/~wilken/code/gmpfrxx.
- MPFUN90. Similarto ARPREC, but is written entirely in Fortran-90 and provides only a Fortran-90 interface. Available at http://crd.lbl.gov/~dhbailey/mpdist.
- QD. This package perform “double-double” (approx. 31 digits) and “quad-double” (approx. 62 digits) arithmetic. C++ and Fortran-90 high-level interfaces. Available at http://crd.lbl.gov/~dhbailey/mpdist.

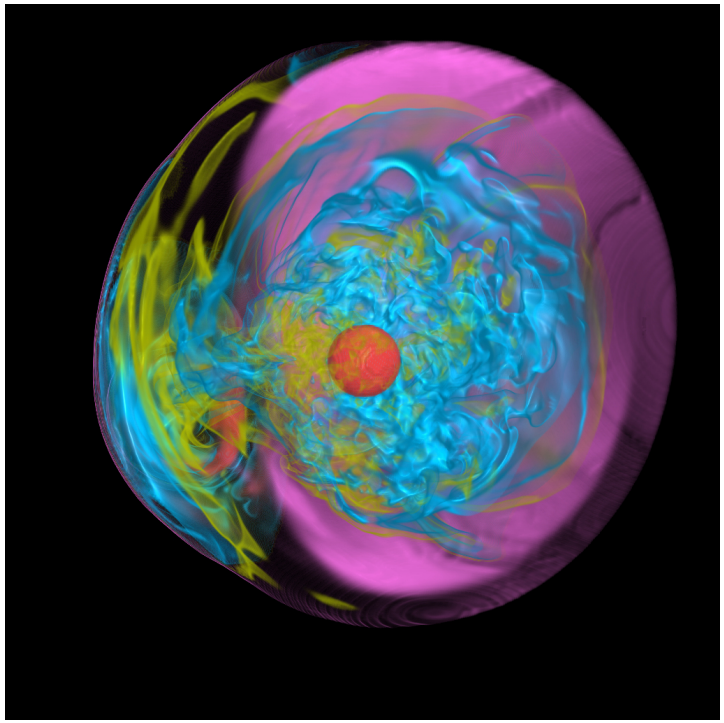
All of these packages greatly increase run time – from ~5X for double-double to ~1000X for arbitrary precision with 1000 digits.

Some applications of high-precision arithmetic

- Supernova simulations (32-64 digits).
- Climate modeling (32 digits).
- Planetary orbit calculations (32 digits).
- Coulomb *n*-body atomic system simulations (32-120 digits).
- Schrodinger solutions for lithium and helium atoms (32 digits).
- Electromagnetic scattering theory (32-100 digits).
- Scattering amplitudes of quarks, gluons and bosons (32 digits).
- Theory of nonlinear oscillators (64 digits).
- Experimental mathematics (100-20,000 digits).
- Evaluating orthogonal polynomials (32-64 digits).
- High-precision ordinary differential equations (100-550 digits).
- Detecting “strange” nonchaotic attractors (32 digits).
- Ising integrals from mathematical physics (100-1000 digits).
- Discrete dynamical systems (32 digits).

Supernova simulations

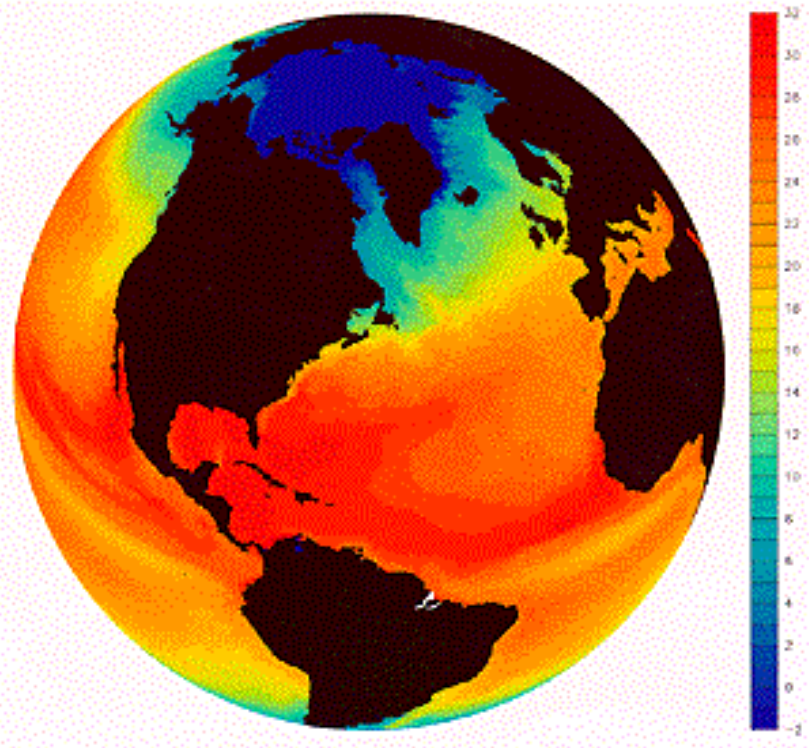
- Researchers at LBNL are using QD to solve for non-local thermodynamic equilibrium populations of iron and other atoms in the atmospheres of supernovas.
- Iron may exist in several species, so it is necessary to solve for all species simultaneously.
- Since the relative population of any state from the dominant state is proportional to the exponential of the ionization energy, the dynamic range of these values can be very large.
- The quad-double portion now dominates the entire computation.



P. H. Hauschildt and E. Baron, "The Numerical Solution of the Expanding Stellar Atmosphere Problem," *Journal Computational and Applied Mathematics*, vol. 109 (1999), pg. 41-63.

Climate modeling

- Climate and weather simulations are fundamentally chaotic – if microscopic changes are made to the current state, soon the future state is quite different.
- In practice, computational results are altered even if minor changes are made to the code or the system.
- This numerical variation is a major nuisance for code maintenance.
- He and Ding of LBNL found that by using double-double arithmetic to implement a key inner product loop, most of this numerical variation disappeared.



Yun He and Chris Ding, "Using Accurate Arithmetics to Improve Numerical Reproducibility and Stability in Parallel Applications," *Journal of Supercomputing*, vol. 18, no. 3 (Mar 2001), pg. 259-277.

Coulomb N-body atomic system simulations

- Alexei Frolov of Queen's University in Canada has used MPFUN90 to solve a generalized eigenvalue problem that arises in Coulomb n-body interactions.
- Matrices are typically 5,000 x 5,000 and are very nearly singular.
- Frolov has also computed elements of the Hamiltonian matrix and the overlap matrix in four- and five-body systems.
- These computations typically require 120-digit arithmetic.

"We can consider and solve the bound state few-body problems which have been beyond our imagination even four years ago." – Frolov

A. M. Frolov and DHB, "Highly Accurate Evaluation of the Few-Body Auxiliary Functions and Four-Body Integrals," *Journal of Physics B*, vol. 36, no. 9 (14 May 2003), pg. 1857-1867.

Experimental math: Discovering new mathematical results by computer

- Compute various mathematical entities (limits, infinite series sums, definite integrals) to high precision, typically 100-1000 digits.
- Use algorithms such as PSLQ to recognize these entities in terms of well-known mathematical constants.
- When results are found experimentally, seek to find formal mathematical proofs of the discovered relations.

Many results have recently been found using this methodology, both in pure mathematics and in mathematical physics.

"If mathematics describes an objective world just like physics, there is no reason why inductive methods should not be applied in mathematics just the same as in physics." – Kurt Godel

Mathematics → Computer science → Scientific computing → Mathematics

The PSLQ integer relation algorithm

Let (x_n) be a given vector of real numbers. An integer relation algorithm finds integers (a_n) such that

$$a_1 x_1 + a_2 x_2 + \cdots + a_n x_n = 0$$

(or within "epsilon" of zero, where epsilon = 10^{-p} and p is the precision).

At the present time the "PSLQ" algorithm of mathematician-sculptor Helaman Ferguson is the most widely used integer relation algorithm. It was named one of ten "algorithms of the century" by *Computing in Science and Engineering*.

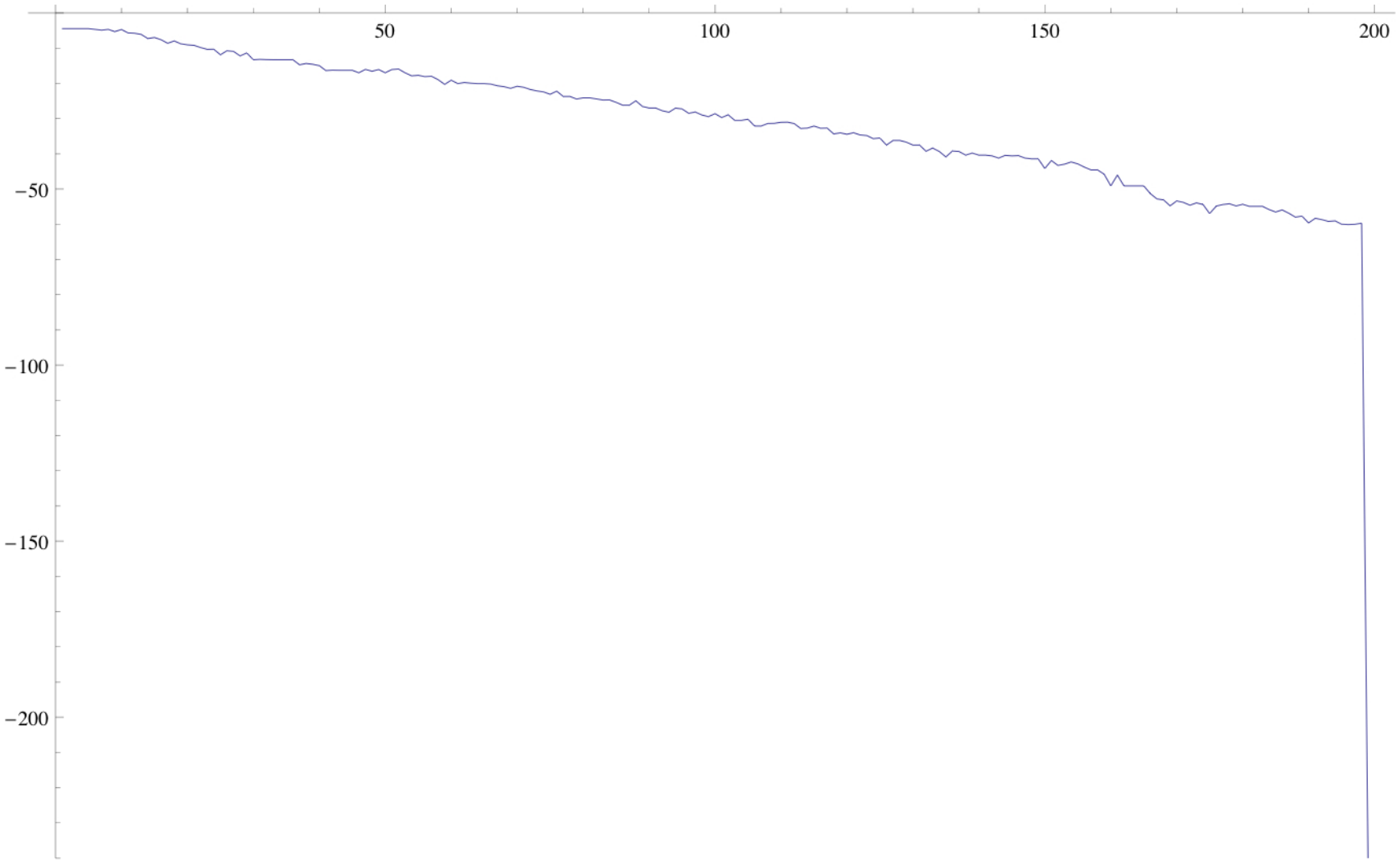
PSLQ (or any other integer relation scheme) requires very high precision (at least $n*d$ digits, where d is the size in digits of the largest a_k), both in the input data and in the operation of the algorithm.

1. H. R. P. Ferguson, DHB and S. Arno, "Analysis of PSLQ, An Integer Relation Finding Algorithm," *Mathematics of Computation*, vol. 68, no. 225 (Jan 1999), pg. 351-369.
2. DHB and D. J. Broadhurst, "Parallel Integer Relation Detection: Techniques and Applications," *Mathematics of Computation*, vol. 70, no. 236 (Oct 2000), pg. 1719-1736.

PSLQ, continued

- PSLQ constructs a sequence of integer-valued matrices B_n that reduces the vector $y = x * B_n$, until either the relation is found (as one of the columns of B_n), or else precision is exhausted.
- At the same time, PSLQ generates a steadily growing bound on the size of any possible relation.
- When a relation is found, the size of smallest entry of the vector y suddenly drops to roughly “epsilon” (i.e. 10^{-p}, where p is the number of digits of precision).
- The size of this drop can be viewed as a “confidence level” that the relation is real and not merely a numerical artifact -- a drop of 20+ orders of magnitude almost always indicates a real relation.

Decrease of $\log_{10}(\min |x_i|)$ in PSLQ



PSLQ discovery: The BBP formula for Pi

In 1996, this new formula for π was found using a PSLQ program:

$$\pi = \sum_{k=0}^{\infty} \frac{1}{16^k} \left(\frac{4}{8k+1} - \frac{2}{8k+4} - \frac{1}{8k+5} - \frac{1}{8k+6} \right)$$

This formula permits one to compute binary (or hexadecimal) digits of π beginning at an arbitrary starting position, using a very simple scheme that can run on any system, using only standard 64-bit or 128-bit arithmetic.

Recently it was proven that no base-*n* formulas of this type exist for π, except $n = 2^m$.

1. DHB, P. B. Borwein and S. Plouffe, "On the rapid computation of various polylogarithmic constants," *Mathematics of Computation*, vol. 66, no. 218 (Apr 1997), pg. 903-913.

2. J. M. Borwein, W. F. Galway and D. Borwein, "Finding and excluding b-ary Machin-type BBP formulae," *Canadian Journal of Mathematics*, vol. 56 (2004), pg 1339-1342.

Some other new BBP-type formulas discovered using high-precision PSLQ

$$\pi^2 = \frac{1}{8}\sum_{k=0}^{\infty}\frac{1}{64^k}\left(\frac{144}{(6k+1)^2}-\frac{216}{(6k+2)^2}-\frac{72}{(6k+3)^2}-\frac{54}{(6k+4)^2}+\frac{9}{(6k+5)^2}\right)$$

$$\pi^2 = \frac{2}{27}\sum_{k=0}^{\infty}\frac{1}{729^k}\left(\frac{243}{(12k+1)^2}-\frac{405}{(12k+2)^2}-\frac{81}{(12k+4)^2}-\frac{27}{(27k+5)^2}\right.$$
$$\left.-\frac{72}{(12k+6)^2}-\frac{9}{(12k+7)^2}-\frac{9}{(12k+8)^2}-\frac{5}{(12k+10)^2}+\frac{1}{(12k+11)^2}\right)$$

$$\zeta(3) = \frac{1}{1792}\sum_{k=0}^{\infty}\frac{1}{2^{12k}}\left(\frac{6144}{(24k+1)^3}-\frac{43008}{(24k+2)^3}+\frac{24576}{(24k+3)^3}+\frac{30720}{(24k+4)^3}\right.$$
$$-\frac{1536}{(24k+5)^3}+\frac{3072}{(24k+6)^3}+\frac{768}{(24k+7)^3}-\frac{3072}{(24k+9)^3}-\frac{2688}{(24k+10)^3}$$
$$-\frac{192}{(24k+11)^3}-\frac{1536}{(24k+12)^3}-\frac{96}{(24k+13)^3}-\frac{672}{(24k+14)^3}-\frac{384}{(24k+15)^3}$$
$$+\frac{24}{(24k+17)^3}+\frac{48}{(24k+18)^3}-\frac{12}{(24k+19)^3}+\frac{120}{(24k+20)^3}+\frac{48}{(24k+21)^3}$$
$$\left.-\frac{42}{(24k+22)^3}+\frac{3}{(24k+23)^3}\right)$$

where ζ is the Riemann zeta function.

DHB, "A compendium of BBP-type formulas," 2010, http://crd.lbl.gov/~dhbailey/dhbpapers/bbp-formulas.pdf.

A connection between BBP formulas and digit randomness

Let {} denote fractional part. Consider the sequence defined by $x_0 = 0$,

$$x_n = \left\{ 16x_{n-1} + \frac{120n^2 - 89n + 16}{512n^4 - 1024n^3 + 712n^2 - 206n + 21} \right\}$$

We showed that π is 16-normal ("random" base-16 digits in a certain sense) if and only if this sequence is equidistributed in the unit interval.

Further, we proved that the following mathematical constant is 2-normal:

$$\begin{aligned} \alpha_{2,3} &= \sum_{n=1}^{\infty} \frac{1}{3^n 2^{3^n}} \\ &= 0.041883680831502985071252898624571682426096\ldots_{10} \\ &= 0.0ab8e38f684bda12f684bf35ba781948b0fcd6e9e0\ldots_{16} \end{aligned}$$

1. D. H. Bailey and R. E. Crandall, "On the Random Character of Fundamental Constant Expansions," *Experimental Mathematics*, vol. 10, no. 2 (Jun 2001), pg. 175-190.
2. D. H. Bailey and R. E. Crandall, "Random Generators and Normal Numbers," Experimental Mathematics, vol. 11, no. 4 (2002), pg. 527-546.
3. D. H. Bailey, "A Pseudo-Random Number Generator Based on Normal Numbers," manuscript, Dec 2004, http://crd.lbl.gov/~dhbailey/dhbpapers/normal-random.pdf.

Discovering AND proving new mathematical formulas by computer

For certain types of mathematical formulas, we can discover them using PSLQ, then prove them using the Wilf-Zeilberger algorithm.

Here is one example of a new mathematical result that was both discovered and proven by computer:

$$\sum_{n=0}^{\infty} \zeta(2n+2)x^{2n} = \frac{1-\pi x\cot(\pi x)}{2x^2} = \sum_{k=1}^{\infty}\frac{1}{k^2-x^2}$$

$$= 3\sum_{k=1}^{\infty}\frac{1}{k^2\binom{2k}{k}(1-x^2/k^2)}\prod_{m=1}^{k-1}\left(\frac{1-4\,x^2/m^2}{1-x^2/m^2}\right)$$

DHB, J. M. Borwein and D. M. Bradley, "Experimental Determination of Apery-Like Identities for Zeta(2n+2)," *Experimental Mathematics*, vol. 15 (2006), pg. 281-289.

Orthogonal polynomials in Sobolev spaces

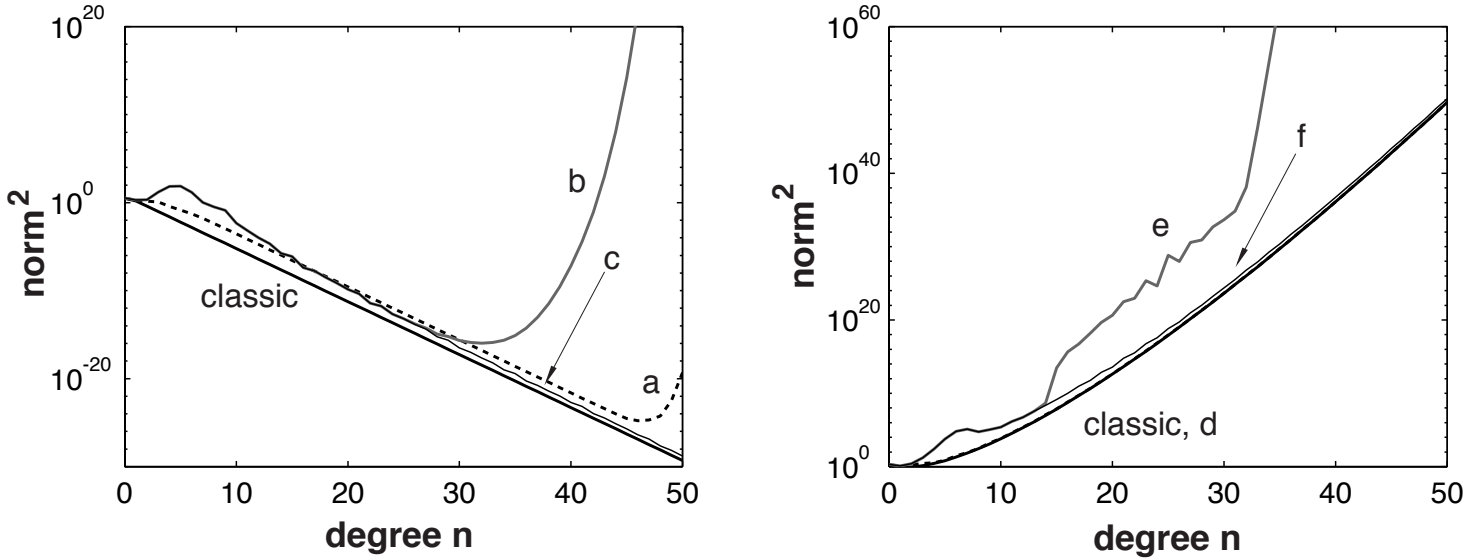
Given a set of K evaluation points, a set $\{r_1, \ldots, r_K\}$ of indices of the maximum order of derivatives at the evaluation points, and non-negative coefficients λ_{ji}, define

$$\langle p, q \rangle_W = \int_R p(x)\, q(x)\, \mathrm{d}\mu_0(x) + \sum_{j=1}^{K} \sum_{i=0}^{r_j} \lambda_{ji}\, p^{(i)}(c_j)\, q^{(i)}(c_j), \qquad \lambda_{ji} \geq 0$$

The formulas that have been derived to obtain basis coefficients for this metric are quite unstable numerically, in part because of the need to compute derivatives of polynomials at support points of discrete measures. Derivative calculations by themselves often lead to serious rounding errors.

1. R. Barrio, B. Melendo and S. Serrano, "Generation and evaluation of orthogonal polynomials in discrete Sobolev spaces I. Algorithms," *Journal of Computational and Applied Mathematics*, vol. 181 (2005), 280-298.
2. R. Barrio and S. Serrano, "Generation and evaluation of orthogonal polynomials in discrete Sobolev spaces II. Numerical stability," *Journal of Computational and Applied Mathematics*, vol. 181 (2005), 299-320.

Computing norms of orthogonal polynomials using 128-bit and 256-bit arithmetic

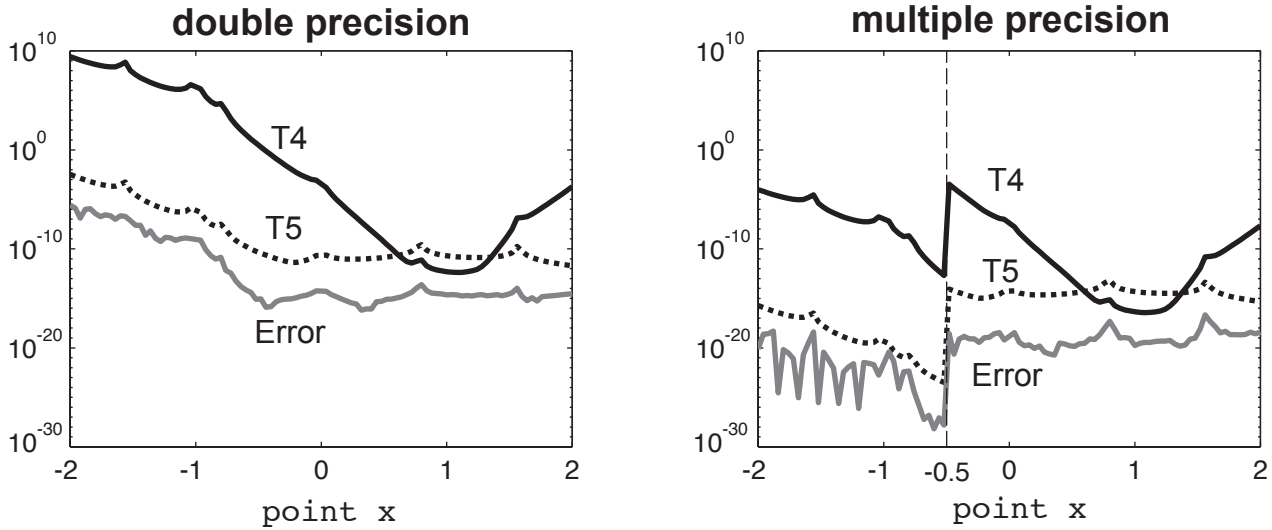


Evaluation (degree 0 to 50) of the square of the L_2-norm of four families of Sobolev orthogonal polynomials compared with the associated classical orthogonal polynomials.

On the left, Chebyshev-Sobolev polynomials with: (a) one mass point (1.5) up to first derivative, lambda = 1/10, using 128 bits, (b) three mass points (-1, 0, 0.5) up to third derivative, lambda = 1/10, using 128 bits, and (c) the same as (b) but using 256 bits.

On the right, Hermite-Sobolev polynomials with: (d) one mass point (1.5) up to first derivative, lambda = 1/10, using 128 bits, (e) three mass points (-1, 0, 0.5) up to 5th derivative, lambda = 1/10, using 128 bits, and (f) the same as (e) but using 256 bits.

Error bounds in Chebyshev-Sobolev: Double vs multiple precision



Behavior of the theoretical error bounds (T4 a backward error bound and T5 for the running error bound) and the relative error in the double- and multiple-precision evaluation of the Chebyshev-Sobolev approximation of degree 50 of the function $f(x) = (x+1)^2 \sin(4x)$, where the discrete Sobolev measure have one mass point (c = 1) up to first derivative in the discrete part of the inner product.

In the figure on the left we use double precision and on the right multiple-precision (on the left of the vertical line we use 96 bits on the mantissa and 64 on the right part).

Taylor's method for the solution of ODEs

Taylor's method is one of the oldest numerical schemes for solving ODEs, but in recent years has re-emerged as the method of choice in the computational dynamics community because of speed to convergence.

Consider the initial value problem y' = f(t, y). The solution at time t = ti is:

$$
\begin{aligned}
y(t_0) &=: y_0, \\
y(t_i) &\simeq y_{i-1} + f(t_{i-1}, y_{i-1})\, h_i + \cdots + \frac{1}{n!} \frac{d^{n-1} f(t_{i-1}, y_{i-1})}{dt^{n-1}} h_i^n \\
&=: y_i
\end{aligned}
$$

The Taylor coefficients here may be found using automatic differentiation methods.

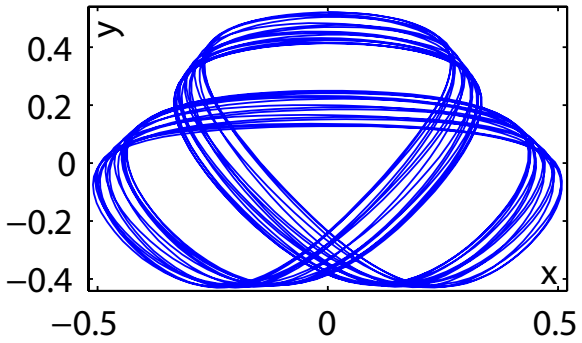
One disadvantage with Taylor's method is that it often requires high-precision arithmetic. But when implemented with high-precision, Taylor's method typically gives superior results, compared with other available schemes (see next talk.)

A. Abad, R. Barrio, F. Blesa and M. Rodriguez, "TIDES: a Taylor series Integrator for Differential EquationS," preprint, 2010.

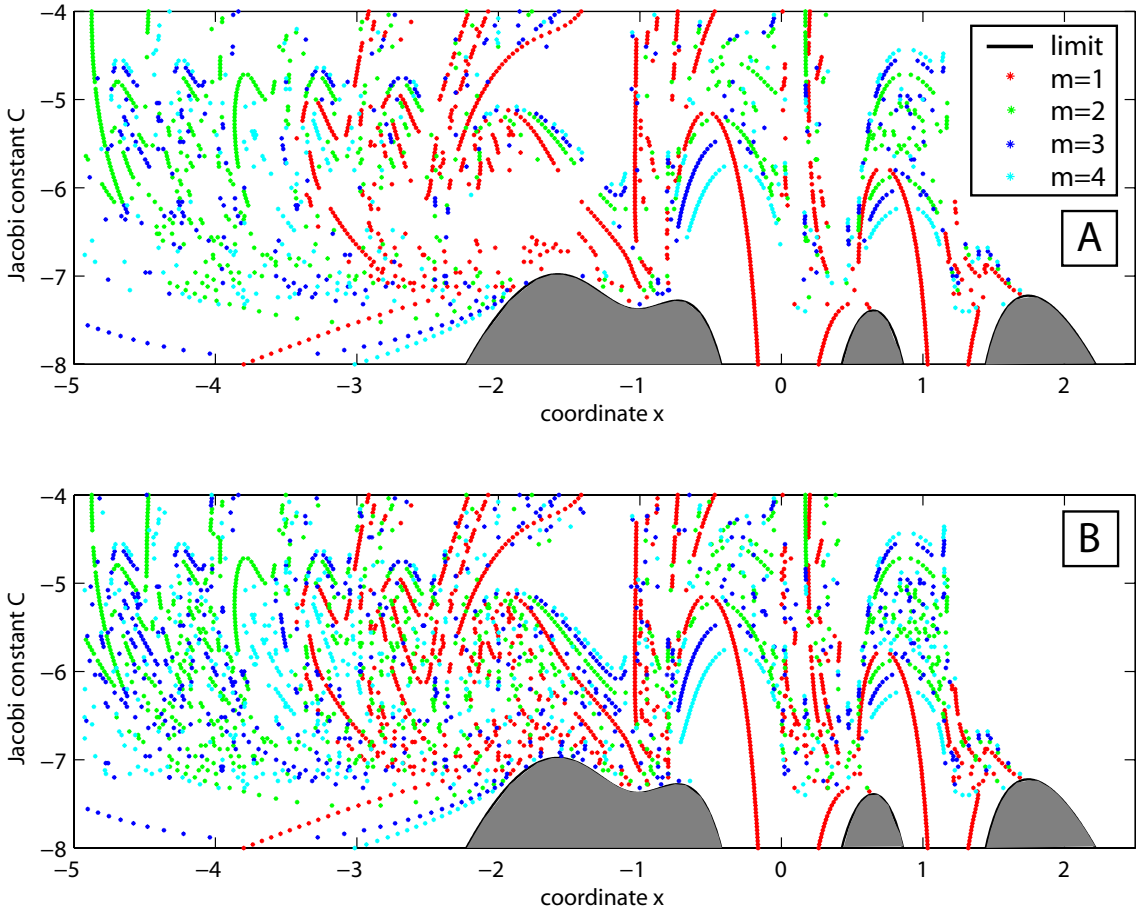
Taylor's method with high-precision arithmetic

Precision	Tol	TIDES (Taylor) CPU Time	TIDES (Taylor) RelErr	dop853 CPU Time	dop853 RelErr
dp	10^{-10}	0.53E−02	0.201E−10	0.34E−02	0.205E−06
dp	10^{-15}	0.12E−01	0.345E−13	0.15E−01	0.113E−11
qp	10^{-20}	0.30E+00	0.300E−20	0.30E+01	0.102E−17
qp	10^{-25}	0.61E+00	0.165E−26	0.12E+02	0.325E−23
mpf90	10^{-32}	0.13E+01	0.782E−29		
mpf90	10^{-64}	0.89E+01	0.144E−65		
mpf90	10^{-128}	0.74E+02	0.432E−131		

CPU time and final error using the Hairer-Wanner code (dop853) and a Taylor method (TIDES) with VSVO formulation for the Henon-Heiles problem, using different precision levels: double precision (dp) for tolerance levels 10^{-10}, 10^{-15}, quadruple precision (qp) for tolerance levels 10^{-20}, 10^{-25}, and multiple precision (mpf90) for tolerance levels 10^{-32}, 10^{-64}, 10^{-128}. The figure at the right shows the computed orbit (an orbit on a KAM tori).



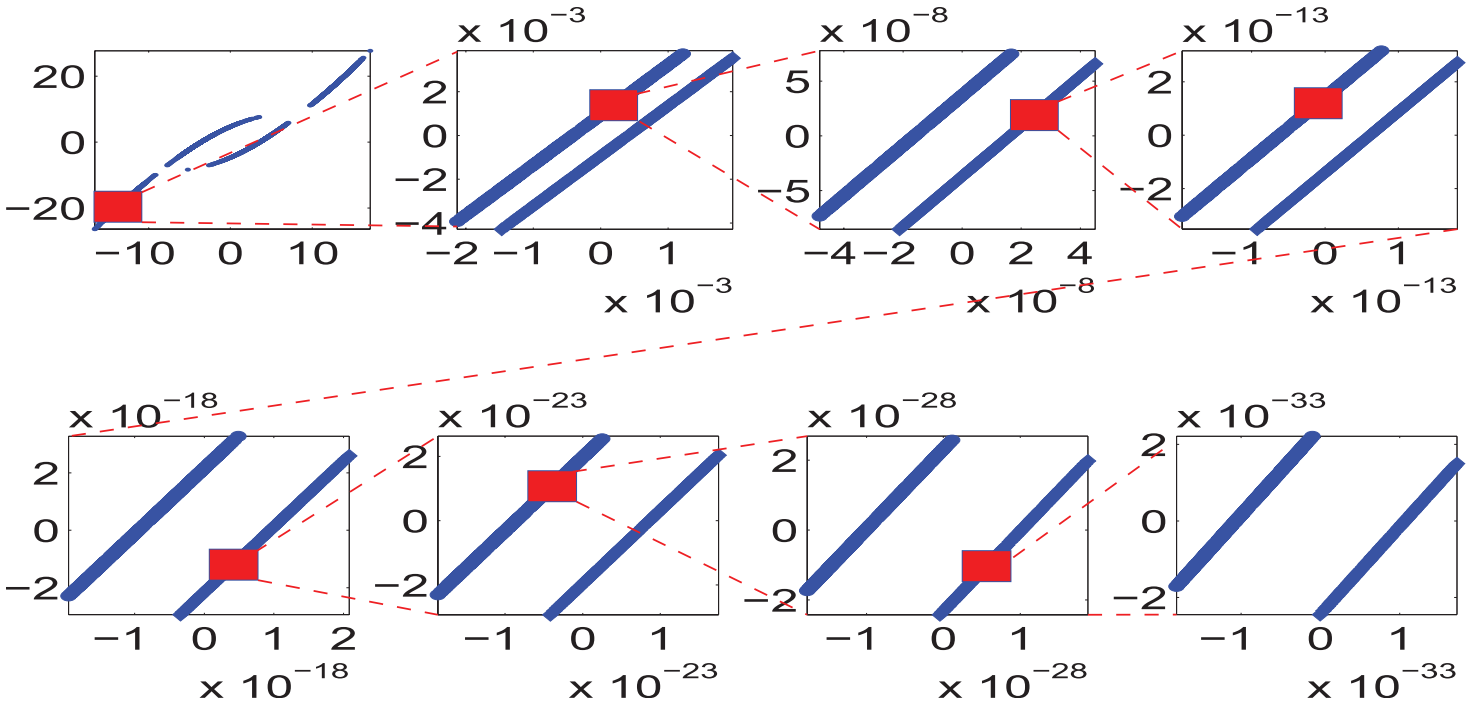
Computing the "skeleton" of periodic orbits



Symmetric periodic orbits (m denotes the multiplicity of the periodic orbit) in the most chaotic zone of the (7+2) ring problem using double (A) and quadruple (B) precision.

R. Barrio and F. Blesa, "Systematic search of symmetric periodic orbits in 2DOF Hamiltonian systems," *Chaos, Solitons and Fractals*, vol. 41 (2009), 560-582.

Fractal properties of Lorenz attractors



On the first plot, the intersection of an arbitrary trajectory on the Lorenz attractor with the section z = 27. The plot shows a rectangle in the x-y plane. All later plots zoom in on a tiny region (too small to be seen by the unaided eye) at the center of the red rectangle of the preceding plot to show that what appears to be a line is in fact not a line. Very high precision (hundreds of digits) are required for these results (see last talk of session).

1. D. Viswanath, "The fractal property of the Lorenz attractor," *Journal of Physics D*, vol. 190 (2004), 115-128.
2. D. Viswanath and S. Sahutoglu, "Complex singularities and the Lorenz attractor," *SIAM Review*, to appear.

High-precision tanh-sinh quadrature

Given $f(x)$ defined on (-1,1), define $g(t) = \tanh(\pi/2 \sinh t)$. Then setting $x = g(t)$ yields

$$\int_{-1}^{1} f(x)\,dx \;=\; \int_{-\infty}^{\infty} f(g(t))g'(t)\,dt \;\approx\; h \sum_{j=-N}^{N} w_j f(x_j),$$

where $x_j = g(hj)$ and $w_j = g'(hj)$. Since $g'(t)$ goes to zero very rapidly for large t, the product $f(g(t))\,g'(t)$ typically is a nice bell-shaped function for which the Euler-Maclaurin formula implies that the simple summation above is remarkably accurate. Reducing h by half typically doubles the number of correct digits.

For our applications, we have found that tanh-sinh is the best general-purpose integration scheme for functions with vertical derivatives or singularities at endpoints, or for any function at very high precision (> 1000 digits). Otherwise we use Gaussian quadrature.

1. DHB, Xiaoye S. Li and Karthik Jeyabalan, "A Comparison of Three High-Precision Quadrature Schemes," *Experimental Mathematics*, vol. 14 (2005), no. 3, pg. 317-329.
2. H. Takahasi and M. Mori, "Double Exponential Formulas for Numerical Integration," Publications of RIMS, Kyoto University, vol. 9 (1974), pg. 721–741.

Ising integrals from mathematical physics

We recently applied our methods to study three classes of integrals that arise in the Ising theory of mathematical physics – D_n and two others:

$$C_n := \frac{4}{n!}\int_0^\infty \cdots \int_0^\infty \frac{1}{\left(\sum_{j=1}^n (u_j + 1/u_j)\right)^2} \frac{\mathrm{d}u_1}{u_1} \cdots \frac{\mathrm{d}u_n}{u_n}$$

$$D_n := \frac{4}{n!}\int_0^\infty \cdots \int_0^\infty \frac{\prod_{i<j} \left(\frac{u_i - u_j}{u_i + u_j}\right)^2}{\left(\sum_{j=1}^n (u_j + 1/u_j)\right)^2} \frac{\mathrm{d}u_1}{u_1} \cdots \frac{\mathrm{d}u_n}{u_n}$$

$$E_n = 2\int_0^1 \cdots \int_0^1 \left(\prod_{1\le j<k\le n} \frac{u_k - u_j}{u_k + u_j}\right)^2 \mathrm{d}t_2\, dt_3 \cdots \mathrm{d}t_n$$

where in the last line $u_k = t_1\, t_2 \ldots t_k$.

DHB, J. M. Borwein and R. E. Crandall, "Integrals of the Ising class," *Journal of Physics A: Mathematical and General*, vol. 39 (2006), pg. 12271-12302.

Limiting value of C_n: What is this number?

The C_n numerical values appear to approach a limit. For instance,

$$C_{1024} = 0.63047350337438679612204019271087890435458707871273234\ldots$$

What is this limit? We copied the first 50 digits of this numerical value into the online Inverse Symbolic Calculator (ISC):
http://ddrive.cs.dal.ca/~isc or **http://carma-lx1.newcastle.edu.au:8087/**

The result was:

$$\lim_{n\to\infty} C_n \quad = \quad 2e^{-2\gamma}$$

where gamma denotes Euler's constant. Finding this limit led us to the asymptotic expansion and made it clear that the integral representation of C_n is fundamental.

Other Ising integral evaluations found using high-precision PSLQ

$$
\begin{aligned}
D_2 &= 1/3 \\
D_3 &= 8 + 4\pi^2/3 - 27\,\mathrm{L}_{-3}(2) \\
D_4 &= 4\pi^2/9 - 1/6 - 7\zeta(3)/2 \\
E_2 &= 6 - 8\log 2 \\
E_3 &= 10 - 2\pi^2 - 8\log 2 + 32\log^2 2 \\
E_4 &= 22 - 82\zeta(3) - 24\log 2 + 176\log^2 2 - 256(\log^3 2)/3 \\
&\quad +16\pi^2\log 2 - 22\pi^2/3 \\
E_5 &\stackrel{?}{=} 42 - 1984\,\mathrm{Li}_4(1/2) + 189\pi^4/10 - 74\zeta(3) - 1272\zeta(3)\log 2 \\
&\quad +40\pi^2\log^2 2 - 62\pi^2/3 + 40(\pi^2\log 2)/3 + 88\log^4 2 \\
&\quad +464\log^2 2 - 40\log 2
\end{aligned}
$$

where ζ is the Riemann zeta function and $\mathrm{Li}_n(x)$ is the polylog function. D_2, D_3 and D_4 were originally provided to us by mathematical physicist Craig Tracy, who hoped that our tools could help identify D_5.

The Ising integral E_5

We were able to reduce E_5, which is a 5-D integral, to an extremely complicated 3-D integral.

We computed this integral to 250-digit precision, using a highly parallel, high-precision 3-D quadrature program. Then we used a PSLQ program to discover the evaluation given on the previous page.

We also computed D_5 to 500 digits, but were unable to identify it. The digits are available if anyone wishes to further explore this question.

$$
\begin{aligned}
E_5 \;=\; & \int_0^1\int_0^1\int_0^1 \left[2(1-x)^2(1-y)^2(1-xy)^2(1-z)^2(1-yz)^2(1-xyz)^2\right. \\
& \left(-\left[4(x+1)(xy+1)\log(2)\left(y^5z^3x^7-y^4z^2(4(y+1)z+3)x^6-y^3z\left(\left(y^2+1\right)z^2+4(y+\right.\right.\right.\right. \\
& 1)z+5\left.\right)x^5+y^2\left(4y(y+1)z^3+3\left(y^2+1\right)z^2+4(y+1)z-1\right)x^4+y\left(z\left(z^2+4z\right.\right. \\
& \left.+5\right)y^2+4\left(z^2+1\right)y+5z+4\left.\right)x^3+\left(\left(-3z^2-4z+1\right)y^2-4zy+1\right)x^2-(y(5z+4) \\
& \left.+4)x-1)\right]/\left[(x-1)^3(xy-1)^3(xyz-1)^3\right]+\left[3(y-1)^2y^4(z-1)^2z^2(yz\right. \\
& -1)^2x^6+2y^3z\left(3(z-1)^2z^3y^5+z^2\left(5z^3+3z^2+3z+5\right)y^4+(z-1)^2z\right. \\
& \left(5z^2+16z+5\right)y^3+\left(3z^5+3z^4-22z^3-22z^2+3z+3\right)y^2+3\left(-2z^4+z^3+2\right. \\
& \left.z^2+z-2\right)y+3z^3+5z^2+5z+3\left.\right)x^5+y^2\left(7(z-1)^2z^4y^6-2z^3\left(z^3+15z^2\right.\right. \\
& \left.+15z+1\right)y^5+2z^2\left(-21z^4+6z^3+14z^2+6z-21\right)y^4-2z\left(z^5-6z^4-27z^3\right. \\
& \left.-27z^2-6z+1\right)y^3+\left(7z^6-30z^5+28z^4+54z^3+28z^2-30z+7\right)y^2-2\left(7z^5\right. \\
& \left.+15z^4-6z^3-6z^2+15z+7\right)y+7z^4-2z^3-42z^2-2z+7\left.\right)x^4-2y\left(z^3\left(z^3\right.\right. \\
& \left.-9z^2-9z+1\right)y^6+z^2\left(7z^4-14z^3-18z^2-14z+7\right)y^5+z\left(7z^5+14z^4+3\right. \\
& \left.z^3+3z^2+14z+7\right)y^4+\left(z^6-14z^5+3z^4+84z^3+3z^2-14z+1\right)y^3-3\left(3z^5\right. \\
& \left.+6z^4-z^3-z^2+6z+3\right)y^2-\left(9z^4+14z^3-14z^2+14z+9\right)y+z^3+7z^2+7z \\
& \left.+1\right)x^3+\left(z^2\left(11z^4+6z^3-66z^2+6z+11\right)y^6+2z\left(5z^5+13z^4-2z^3-2z^2\right.\right. \\
& \left.+13z+5\right)y^5+\left(11z^6+26z^5+44z^4-66z^3+44z^2+26z+11\right)y^4+\left(6z^5-4\right. \\
& \left.z^4-66z^3-66z^2-4z+6\right)y^3-2\left(33z^4+2z^3-22z^2+2z+33\right)y^2+\left(6z^3+26\right. \\
& \left.z^2+26z+6\right)y+11z^2+10z+11\left.\right)x^2-2\left(z^2\left(5z^3+3z^2+3z+5\right)y^5+z\left(22z^4\right.\right. \\
& \left.+5z^3-22z^2+5z+22\right)y^4+\left(5z^5+5z^4-26z^3-26z^2+5z+5\right)y^3+\left(3z^4-\right. \\
& \left.22z^3-26z^2-22z+3\right)y^2+\left(3z^3+5z^2+5z+3\right)y+5z^2+22z+5\left.\right)x+15z^2+2z \\
& +2y(z-1)^2(z+1)+2y^3(z-1)^2z(z+1)+y^4z^2\left(15z^2+2z+15\right)+y^2\left(15z^4\right. \\
& \left.-2z^3-90z^2-2z+15\right)+15\left.\right]/\left[(x-1)^2(y-1)^2(xy-1)^2(z-1)^2(yz-1)^2\right. \\
& \left.(xyz-1)^2\right]-\left[4(x+1)(y+1)(yz+1)\left(-z^2y^4+4z(z+1)y^3+\left(z^2+1\right)y^2\right.\right. \\
& -4(z+1)y+4x\left(y^2-1\right)\left(y^2z^2-1\right)+x^2\left(z^2y^4-4z(z+1)y^3-\left(z^2+1\right)y^2\right. \\
& \left.+4(z+1)y+1\right)-1)\log(x+1)\left.\right]/\left[(x-1)^3x(y-1)^3(yz-1)^3\right]-\left[4(y+1)(xy\right. \\
& +1)(z+1)\left(x^2\left(z^2-4z-1\right)y^4+4x(x+1)\left(z^2-1\right)y^3-\left(x^2+1\right)\left(z^2-4z-1\right)\right. \\
& y^2-4(x+1)\left(z^2-1\right)y+z^2-4z-1\left.\right)\log(xy+1)\left.\right]/\left[x(y-1)^3y(xy-1)^3(z-\right. \\
& \left.1)^3\right]-\left[4(z+1)(yz+1)\left(x^3y^5z^7+x^2y^4(4x(y+1)+5)z^6-xy^3\left(\left(y^2+\right.\right.\right. \\
& \left.1\right)x^2-4(y+1)x-3\left.\right)z^5-y^2\left(4y(y+1)x^3+5\left(y^2+1\right)x^2+4(y+1)x+1\right)z^4+ \\
& y\left(y^2x^3-4y(y+1)x^2-3\left(y^2+1\right)x-4(y+1)\right)z^3+\left(5x^2y^2+y^2+4x(y+1)\right. \\
& \left.\left.y+1\right)z^2+((3x+4)y+4)z-1\right)\log(xyz+1)\left.\right]/\left[xy(z-1)^3z(yz-1)^3(xyz-1)^3\right]\left.\right)\left.\right] \\
& /\left[(x+1)^2(y+1)^2(xy+1)^2(z+1)^2(yz+1)^2(xyz+1)^2\right]\,dx\,dy\,dz
\end{aligned}
$$

Recursions in Ising integrals

Consider the 2-parameter class of Ising integrals (which arises in QFT for odd *k*):

$$C_{n,k} = \frac{4}{n!}\int_0^\infty \cdots \int_0^\infty \frac{1}{\left(\sum_{j=1}^n (u_j + 1/u_j)\right)^{k+1}} \frac{\mathrm{d}u_1}{u_1} \cdots \frac{\mathrm{d}u_n}{u_n}$$

After computing 1000-digit numerical values for all *n* up to 36 and all *k* up to 75 (performed on a highly parallel computer system), we discovered (using PSLQ) linear relations in the rows of this array. For example, when *n* = 3:

$$\begin{aligned}
0 &= C_{3,0} - 84C_{3,2} + 216C_{3,4} \\
0 &= 2C_{3,1} - 69C_{3,3} + 135C_{3,5} \\
0 &= C_{3,2} - 24C_{3,4} + 40C_{3,6} \\
0 &= 32C_{3,3} - 630C_{3,5} + 945C_{3,7} \\
0 &= 125C_{3,4} - 2172C_{3,6} + 3024C_{3,8}
\end{aligned}$$

Similar, but more complicated, recursions have been found for all *n*.

1. DHB, D. Borwein, J. M. Borwein and R. Crandall, "Hypergeometric Forms for Ising-Class Integrals," *Experimental Mathematics*, to appear, http://crd.lbl.gov/~dhbailey/dhbpapers/meijer/pdf.

2. J. M. Borwein and B. Salvy, "A Proof of a Recursion for Bessel Moments," *Experimental Mathematics*, vol. 17 (2008), pg. 223-230.

Four hypergeometric evaluations (here $c_{n,k} = n!\, k!\, 2^{-n}\, C_{n,k}$)

$$c_{3,0} = \frac{3\Gamma^6(1/3)}{32\pi 2^{2/3}} = \frac{\sqrt{3}\pi^3}{8}\, {}_3F_2\left(\begin{matrix} 1/2, 1/2, 1/2 \\ 1, 1 \end{matrix} \middle| \frac{1}{4} \right)$$

$$c_{3,2} = \frac{\sqrt{3}\pi^3}{288}\, {}_3F_2\left(\begin{matrix} 1/2, 1/2, 1/2 \\ 2, 2 \end{matrix} \middle| \frac{1}{4} \right)$$

$$c_{4,0} = \frac{\pi^4}{4} \sum_{n=0}^{\infty} \frac{\binom{2n}{n}^4}{4^{4n}} = \frac{\pi^4}{4}\, {}_4F_3\left(\begin{matrix} 1/2, 1/2, 1/2, 1/2 \\ 1, 1, 1 \end{matrix} \middle| 1 \right)$$

$$c_{4,2} = \frac{\pi^4}{64}\left[4\, {}_4F_3\left(\begin{matrix} 1/2, 1/2, 1/2, 1/2 \\ 1, 1, 1 \end{matrix} \middle| 1 \right) - 3\, {}_4F_3\left(\begin{matrix} 1/2, 1/2, 1/2, 1/2 \\ 2, 1, 1 \end{matrix} \middle| 1 \right) \right] - \frac{3\pi^2}{16}$$

DHB, J. M. Borwein, D. Broadhurst and M. L. Glasser, "Elliptic Integral Evaluations of Bessel Moments," *Journal of Physics A: Mathematical and General*, vol. 41 (2008), pg 205203.

2-D integral in Bessel moment study

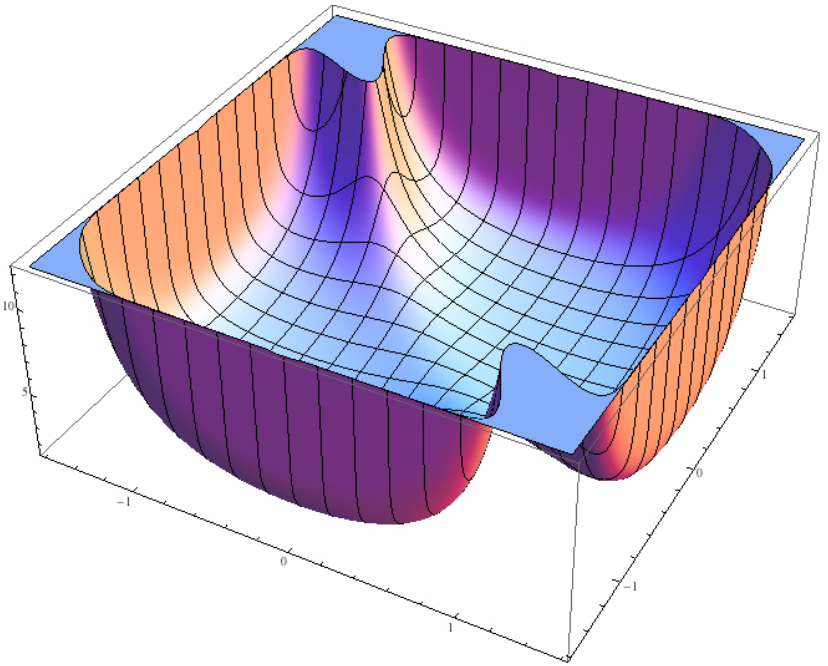
We conjectured (and later proved)

$$c_{5,0} = \frac{\pi}{2} \int_{-\pi/2}^{\pi/2} \int_{-\pi/2}^{\pi/2} \frac{\mathbf{K}(\sin\theta)\,\mathbf{K}(\sin\phi)}{\sqrt{\cos^2\theta\cos^2\phi + 4\sin^2(\theta+\phi)}}\,\mathrm{d}\theta\,\mathrm{d}\phi$$

Here **K** denotes the complete elliptic integral of the first kind

Note that the integrand function has singularities on all four sides of the region of integration.

We were able to evaluate this integral to 120-digit accuracy, using 1024 cores of the "Franklin" Cray XT4 system at LBNL.



Lions-Mercer iterations

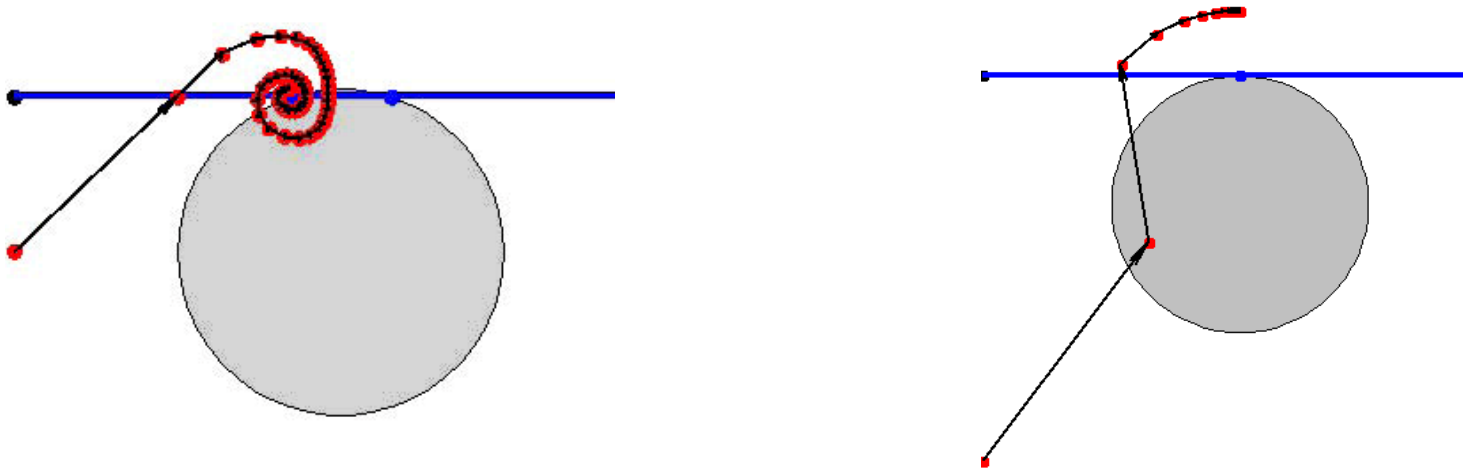
The Lions-Mercer iteration, also known as the Douglas-Rachford or Feinup iteration, is defined by the procedure: reflect, reflect and average:

$$x \mapsto T(x) := \frac{x + R_A\left(R_B(x)\right)}{2}$$

In the simple 2-D case of a horizontal line of height α, we obtain the explicit iteration:

$$x_{n+1} := \cos\theta_n, \quad y_{n+1} := y_n + \alpha - \sin\theta_n, \quad (\theta_n := \arg z_n)$$

For $0 < \alpha < 1$, spiraling is ubiquitous: (α = 0.95 on left, and 1.0 on right):



Exploring iterations using Cinderella

Iterations such as this, as well as many other graphical phenomena, may be explored using the Cinderella online tool: http://www.cinderella.de.

Two applets have been defined, working with Cinderella, for exploring Lions-Mercer iterations:

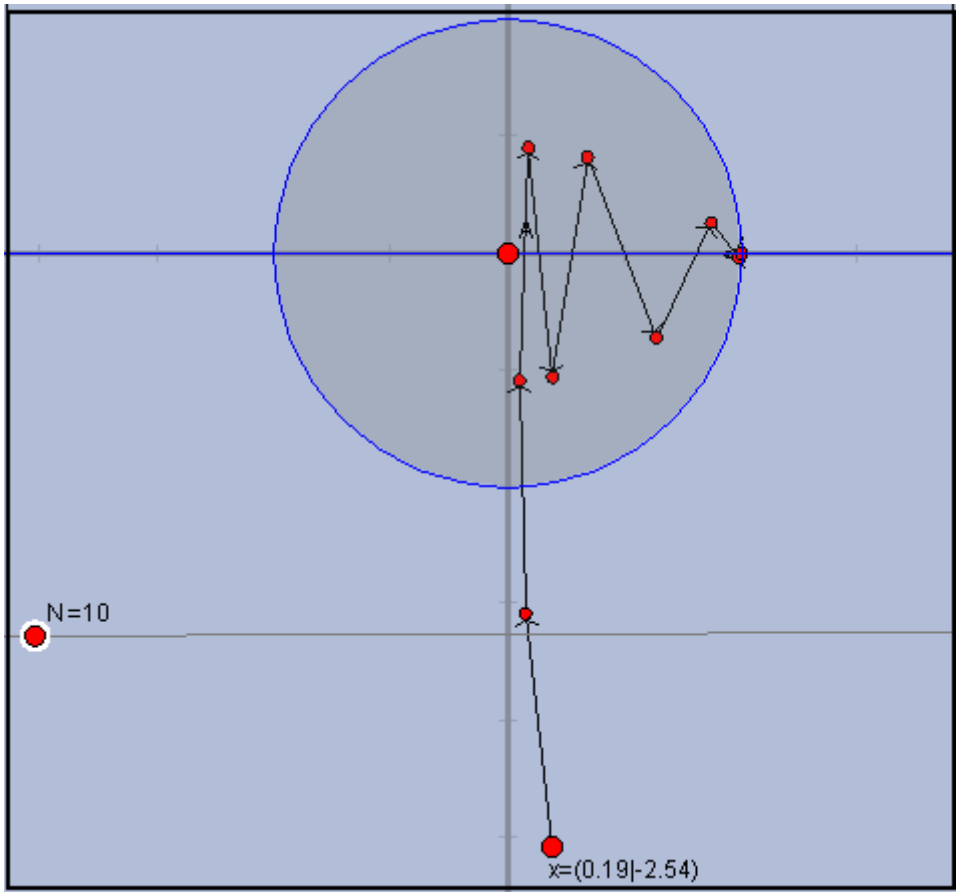
A1. http://users.cs.dal.ca/~jborwein/reflection.html

A2. http://users.cs.dal.ca/~jborwein/expansion.html

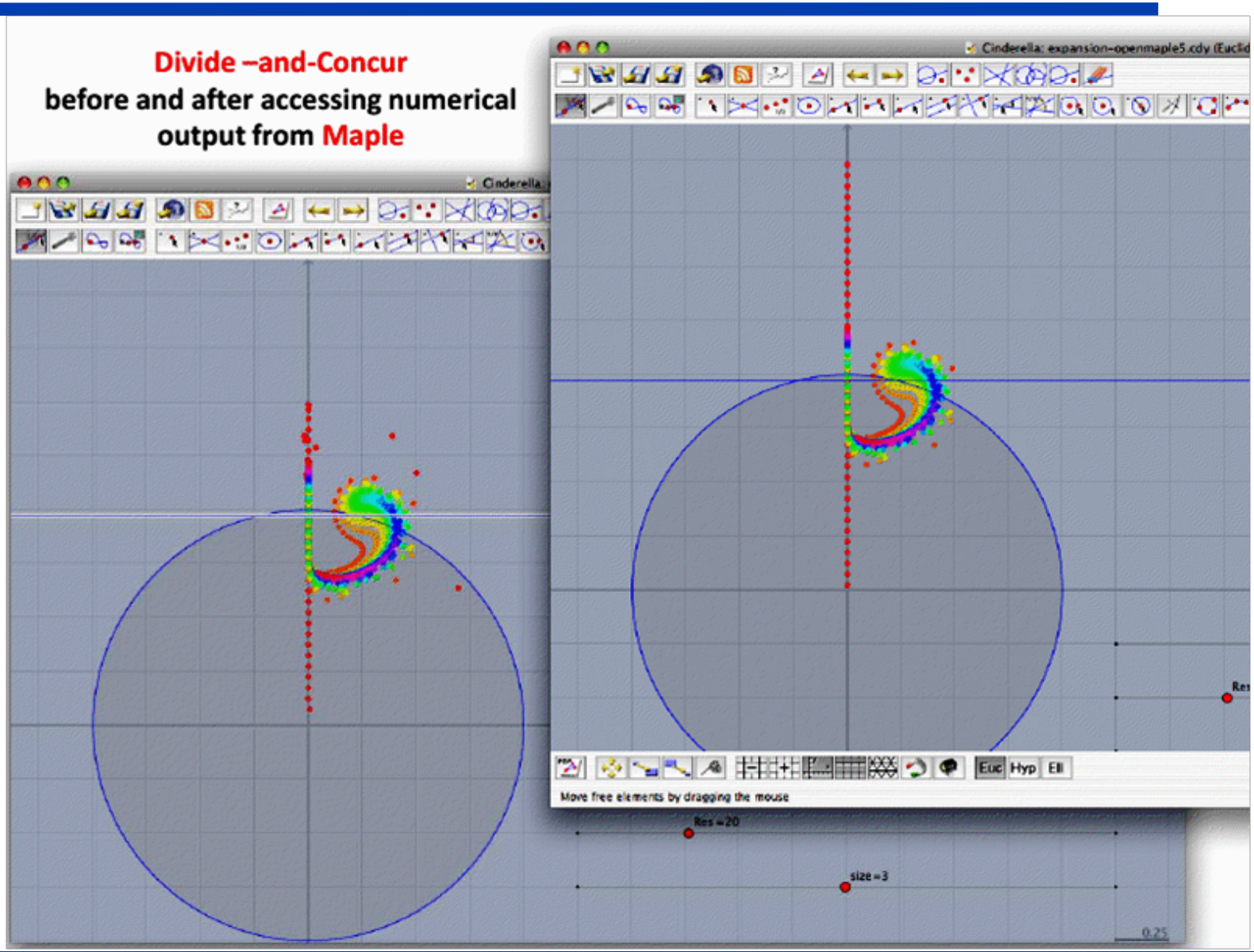
For Applet A1, we observed that (see graphic on next slide):

- As long as the iterate is outside the unit circle the next point is always closer to the origin;
- Once inside the circle the iterate never leaves;
- The angle now oscillates to zero and the trajectory hence converges to (1,0).

Iterations with Applet A1



Iterations with Applet A2: Double vs multiple precision



Summary

- Numerically sensitive features of many algorithms point to the need for numeric precision computation higher than the IEEE 64-bit standard.
- The recent proliferation of very highly parallel systems has the potential of greatly exacerbating numerical difficulties.
- Many real-world applications have now been identified that require double-double or quad-double arithmetic.
- Some research studies, particularly in experimental mathematics and mathematical physics, require hundreds or even thousands of digits.
- Software is now available, mostly for free, to permit conversion of computer programs to use higher precision.
- Such high-precision software facilities are rapidly becoming an indispensible part of a modern high-performance computer system.